تعلم معنا لغة البرمجة

C

شرح مفصل عن اللغة مع مكوناتها وبعض الاكواد البرمجية

الفارعي



الحمد لله رب العالمين،القائل في محكم التنزيل (وعلمك ما لم تكن تعلم، وكان فضل الله عليك عظيماً)،والصلاة والسلام على سيدنا محمد سيد العلماء و سيد الأولين و الآخرين رسول رب العالمين، وعلى آله و صحبه أجمعين .

فمن وجد خطأً فهو مني وما كان فيه من صواب فمن توفيق الله .

والحمد لله رب العالمين.............

Structure programing:

البرمجة الهيكلية

تسمى ب تسلسلية

(oob) object orieuted programming:

البرمجة ارضية التوجه وتسمى ب المفصلة.

نبذه عن اللغات:

1- cpl		
2- Bcpl	1967	ريتشار
3- B	1969	كين
4- C	1978	براين + دنس

ANSI:

Amorican National Stands Institute.

وحدة المعايير الدولية الامريكية.

- لماذا سمية هذه اللغة ب لغة السي؟

لان حرف السي حرف بعد حرف البي حسب المعلومات الأعلاء

واساسيات لغة السي استخدام المكتبة وهي (مخزن في برنامج السي وفيها ملفات الرأسية.

ولمعرفة كتفية الاستخدام اللغة يجب معرفة اصدار هذه اللغة ومعرفة انواعها ومعرفة كتفية التنصيب .

Code source لغة عالية المستوى

Compilerملف مخزن في القرص c ويقوم بتحويل اللغات(ترجمة)

Object file لغة الآلة

Linker labrary c ربط مكتبة

Fxecute file ملف التنفيذ

أنواع البيانات :

إن البيانات التي نتعامل معها إما أرقام او أحرف أو كلمات و الأرقام يمكن أن تكون صحيحة(أي ليس بها علامة عشرية) أو حقيقية أي بها علامة عشرية.

و الحروف يمكن أن تكون حرف واحد أو أكثر من حرف و هكذا تختلف أنواع البيانات عن بعضها البعض و من الضروري معرفة أنواع البيانات و معرفة كيفية الإعلان عنها و كذلك كيفية استعمالها.

و الجدول التالي يوضح هذه الأنواع و كذلك عدد البايت التي يشغلها كل نوع:

نوع المتغير	طوله بالبايت	المدى المسموح
حرف Char	1	حرف أو رمز واحد
صحيح int	2	-32768 إلى 32768
صحيح طويل long	4	-2014704830648 إلى 2014704830648
حقيقي float	4	E-38 إلى E+38
حقيقي مضاعف double	8	E-308 إلى E+308

و نوضح فيما يلي المقصود بكل هذه الأنواع:

- متغير من نوع حرف أي متغير يصلح لتخزين حرف فقط.
- متغير من نوع صحيح أي متغير يصلح لتخزين رقم صحيح(ليس به علامة عشرية مثل ٥ و ٥٧ و ٥٤٤).
- متغير من نوع صحيح و لكن طويل (long)

أي يستطيع أن يخزن رقم صحيح ضعف المتغير الصحيح العادي و نستعمل هذا النوع إذا كانت الأرقام التي نتعامل معها أكبر من المساحة المخصصة للرقم الصحيح العادي وإلا سنحصل على نتائج خاطئة بالرغم من أن البرنامج سليم.

• متغير حقيقي أي متغير يصلح لتخزين رقم حقيقي(يقبل الكسور العشرية مثل ٣.٣ و ٤٥.٤٤ و ١٤٠.٠٠٩).
• متغير حقيقي مضاعف أي يستطيع أن يخزن رقم حقيقي ضعف المتغير الحقيقي العادي.

تسمية المتغير:

يخضع اسم المتغير لشروط معينة يجب أن تعرفها تجنباً لأخطاء قد تقع فيها و فيما يلي نوضح هذه الشروط:

• يجب أن يبدأ المتغير بحرف ثم يكمل المتغير بعد ذلك حروف أو أرقام و يجب ألاّ يحتوي على علامة خاصة سوى الشرطة التحتية(_).
• من الممكن أن يشتمل اسم المتغير حتى ٣٢ حرف و ما زاد عن ذلك لا يلتفت إليه مترجم اللغة.

يختلف عن St• يفرّق المترجم بين الحروف الصغيرة و الكبيرة فالمتغير فإذا استعملا في البرنامج يعتبر هما البرنامج متغيرين.stالمتغير

يجب ألاّ يكون المتغير باسم كلمة من الكلمات المحجوزة في اللغة.

int, return:الإعلان عن المتغيرات

فيمكن أن يتم الإعلان عن C++\Cإذا كنت تستخدم مترجم للغة المتغيرات في أي مكان بالبرنامج و لكن بشرط أن تكون قبل العبارات التي فقط فيجب أن Cتستخدم هذا المتغير أما إذا كنت تستخدم مترجم للغة يكون الإعلان في أول البرنامج لتلافي الأخطاء.

```
int        a;
float
b;
```

Operatorsالمؤثرات :

المؤثرات هي الرموز التي تربط بين المتغيرات و الثوابت لإنشاء علاقة

ما أو معادلة تختلف أنواع المؤثرات باختلاف وظيفة كل مؤثر . و تأخذ الأنواع الآتية:

Arithmetic Operators المؤثرات الحسابية :

و هي علامات الجمع والطرح و القسمة و الضرب وتستخدم مع المتغيرات و الثوابت الرقمية.

Relational Operators مؤثرات المقارنة :

و تستخدم لمقارنة قيمتين لمعرفة هل هما متساويتين أو إحداهما أكبر أو أقل من الأخرى و هكذا. و يوضح الجدول التالي مؤثرات المقارنة و الرموز التي تستخدم بدلاَ ً عنها.

المؤثر	الرمز	مثال	النتيجة
أكبر من	>	10 > 8	1
أصغر من	<	10 < 8	0
يساوي	==	10==8	0
لا يساوي	!=	10 != 8	1
أقل من أو يساوي	<=	10<=8	0
أكبر من أو يساوي	>=	10>=8	1

Logical Operators المؤثرات المنطقية :

تستخدم لتحديد شرط مركب مثل الشرط التالي:

```
if(a==b &&  c==d)
```

تساوي قيمة المتغير c وفي نفس الوقت قيمة المتغير b تساوي معناه إذا كانت قيمة المتغير d

قيمة المتغير a .

يوضح الجدول التالي هذه المؤثرات و الرموز التي تستخدم بدلاً منها:

النتيجة	مثال	الرمز	المؤثر
1	((10>8)&&(9>7))	&&	AND
1	((10<8)||(7<8))	||	OR
1	!(10==8)	!	NOT

Assignment Operators مؤثرات التخصيص

و هي مؤثرات تخزين قيمة في متغير مثل :

=,+=,-
=,*=,/=

و تستخدم لتخزين قيمة في متغير بالاعتماد على القيمة الموجودة في نفس المتغير فمثلاً إذا قمت بتخزين القيمة ٦ في
و أردت مضاعفة القيمة المخزنة يجب أن a=6باستخدام الأمر a المتغير
تساوي ١٢ و لزيادة قيمة a بهذا تصبح قيمة a=a*2تكتب الأمر
وهكذا و هذه الطريقة تستخدم في جميع لغات البرمجة و a=a+1المتغير
بوجود طريقة بجانب الطريقة السابقة موضحة بالجدول Cتتميز لغة
a=6التالي بفرض أن

النتيجة	الطريقة الحديثة	التخصيص التقليدي
11	a+=5	a=a+5
1	a-=5	a=a-5
30	a*=5	a=a*5
2	a/=3	a=a/3
7	a++	a=a+1
5	a--	a=a-1

ملاحظة: هناك فرق بين المؤثر = و المؤثر == حيث أن المؤثر = يستخدم كما سبق في إلحاق قيمة بمتغير أما المؤثر == يستخدم للمقارنة.

مؤثر باقي خارج القسمة %:

يستخدم لمعرفة باقي القسمة(و تستطيع أن تحدد هل الأرقام الموجودة في A=5متغير ما زوجية أم فردية) فمثلاً إذا كانت

C=A%2و كتبت:
ستكون قيمتها ١ وهو باقي قسمة الرقم ٥ على ٢ فإن C

Increment & Decrement مؤثران الزيادة و النقصان
بمقدار A معناه زيادة قيمة المتغير A=A+1من المعروف أن التعبير
بمقدار واحد ولكن A معناه إنقاص قيمة المتغير A=A-1واحد و التعبير
وتقابل Aتوجد صور أخرى مسموح بها لهاتين العمليتان وهي ++
A=A-1 وتقابل A و --A=A+1

دوال :

أن كل دالة مرتبطة بملف توجيه معين حيث يستدعى هذا الملف في أول البرنامج بالعبارة #include فمثلاً الدالة print f()

معرفة بالملف stdio.h وتكتب العبارة في أول البرنامج حتى يتعرف المترجم على الدالة و هكذا مع باقي الدوال.

- دالة الطباعة على الشاشة print f()

ملف التوجيه stdio.h

تستخدم الدالة () printf لطباعة البيانات بجميع أنواعها (int, char, string, float...) على الشاشة فقط.

وتأخذ الدالة عدة صور وكذلك معاملات وأكواد تحدد شكل المخرجات.

وسنوضح فيما يلي مثال لكل صورة مع الشرح

مثال:

Print f ("welcome ")
with Computer Science

وفي هذا المثال يتم طباعة ما بين علامتي التنصيص" "كما هو على الشاشة وبالتالي نحصل على النتيجة التالية:

welcome with Computer Science

مثال:

Print f ("\n Welcome\n with \n Computer Science")

وفي هذا المثال: الكود n\ معناه new line أي سطر جديد وعندما يجد المترجم n\ يترجمها إلى سطر جديد وبالتالي نحصل على النتيجة التالية:

Welcome

with

Computer Science

وهناك أكواد أخرى تؤدي نتائج مختلفة فمثل الكود t\ معناه tab أي مسافة جدولة خالية ويشمل الجدول التالي على الأكواد المستخدمة مع الدالةprint f() والتي تؤدي أشكال خرج مختلفة.

الكود	الاستخدام	مثال
n\	الانتقال لسطر جديد	print f("\n")
t\	نقل المؤشر بعد ٨ مسافات	print f("\t")
a\	إخراج صوت الصافرة	print f("\a")

b\ إرجاع المؤشر مسافة خلفية print f("\b")

xdd\طباعة الحرف المناظر للكود المكتوب بالنظام السادس عشر
print f("\x41 hexadecimal")

ddd\طباعة الحرف المناظر للكود المكتوب بالنظام الثماني (octal")
print f("\101

\\ طباعة الشرط المائلة print f ("\\")

?\ طباعة علامة الاستفهام print f("\?")

'\ طباعة العلامة (') print f("'\")

"\ طباعة علامة التنصيص print f(""\")

• طباعة قيم المتغيرات على الشاشة: لطباعة القيم الموجودة بالمتغيرات نستخدم أكواد معينة لتحدد نوع البيانات المراد طباعتها بالدالة printf ()، انظر للمثال التالي:

```
# Include <stdio.h>
Void main()
{
Int a=5;
Float b=1.5;
Print f ("a= %d" , a);
Print f("\n b = %f" ,b);
```

}

فيكون الناتج

a = 5

b = 1.500000

في هذا المثال عندما يقابل مترجم اللغة العلامة% ينظر إلى الحرف التالي لهذه العلامة ويعتبر هذا الحرف توصيف لقيمة موجودة بعد العلامة، وكل حرف يحدد نوع معين من البيانات ففي هذا المثال نلاحظ:

%d تعني int أي رقم صحيح بينما %f تعني float أي رقم حقيقي .

ويوضح جدول التالي أكواد طباعة أنواع البيانات:

الكود الاستخدام مثال

d% توصيف لمتغير (أو ثابت)رقمي صحيح (int printf("%d",-10

f% توصيف لمتغير(أو ثابت) رقمي حقيقي (float printf("%f",5.7

c% توصيف لمتغير(أو ثابت) char (حرف واحد) (" printf("%c",'a

s% توصيف لعبارة حرفية string (حرق أو أكثر من حرف)("printf("%s","ab

u% توصيف لمتغير (أو ثابت) رقمي صحيح بدون إشارة printf("%u",10)

x% توصيف لمتغير (أو ثابت) بالنظام السادس عشر hex printf("%x",a)

o% توصيف لمتغير(أو ثابت) بالنظام الثماني octal

printf("%o",67)

ملاحظات: يمكن أن تستخدم الأكواد d% أو f% لتحديد عدد الأرقام التي تظهر على الشاشة فمثلاً الصورة ٣.f% يعني طباعة ثلاث أرقام بعد العلامة العشرية فمثلاً الرقم ٥٣٤.٥٦٣٧٨ يظهر بالصورة ٥٣٤.٥٦٤.

- دالة الإدخال العامة () scanf

هي دالة الإدخال الرئيسية التي تسمح بإدخال جميع أنواع البيانات وهي تأخذ نفس المعاملات التي تأخذها الدالة()printf للتعامل مع البيانات والموجودة بالجدول السابق.

والصورة العامة للدالة()scanf هي:

int scanf])

(const char*format [,address،...

مثال التالي يوضح استخدام الدالة()scanf حيث يقوم باستقبال مجموعة قيم مختلفة النوع وطباعتها على الشاشة.

```
# include<stdio.h >
void main (){
int a;
float b;
char ch='Y';
char name[10 ];
printf("\n Enter your name:")
```

```
scanf("%s",name);
printf("a=");
scanf("%d",&a);
printf("b=");
scanf("%f",&b);
printf("\nWlecome    %s",name);
printf("\n\t a = %d",a+1);
printf("\n\t b= %.2f",b);
printf("\n\t c = %c",ch);
{
```

فيكون الناتج :

```
Enter your name : Ahmed
a=12
b=4.5
Welcome   Ahmed
a = 12
b = 4.50
c = Y
```

ونلاحظ أن الدالة ()scanf تستقبل قيمة صحيحة وتخزنها في المتغير a وقيمة حقيقية وتخزنها في المتغير b ولكن ماذا يعني المؤثر & ؟

&a تعني تخزين القيمة الصحيحة في المكان المخزن عنوانه في المتغير a بمعنى أن a يشير إلى عنوان المكان الذي تخزن فيه القيمة وبالتالي العلامة & تجعل المتغير يشير إلى عنوان المكان . ويوضع المؤثر & فقط مع البيانات الصحيحة و الحقيقية و الحرف ولا يوضع مع متغير العبارة الحرفية string .

• دوال إدخال حرف واحد getchar(), getche( ), getch():

بالرغم من أن الدالة scanf() تستقبل جميع أنواع البيانات إلا أن لغة C تشتمل على دوال أخرى تتعامل مع أنواع خاصة من البيانات كالحروف والعبارات الحرفية ونوضح فيما يلي أهم هذه الدوال.

• الدالة getchar () :

ملف التوجيه stdio.h

تستخدم لإدخال حرف واحد ويظهر الحرف على الشاشة بعد الكتابة ولا تسمح بالانتقال إلى الأمر التالي إلا إذا ضغط المستخدم مفتاح الإدخال Enter.

مثال:

```
#include <stdio.h>
void main(){
char a;
printf("Enter Char:")
a=getchar();
printf("%c",a;)
```

```
{
```

فيكون الناتج

```
Enter Char:A
A
```

•الدالة getche():

ملف التوجيه conio.h

تستخدم لإدخال حرف واحد ويظهر هذا الحرف على الشاشة ولكنها تختلف عن الدالة getchar() في أنها لا تحتاج إلى الضغط على مفتاح الإدخال Enter .

مثال:

```
#include <stdio.h>
#include <conio.h>
void main(){
char a;
printf("Enter Char:")
a=getche();
printf("\n%c",a);
{
```

فيكون الناتج

```
Enter Char:C
```

C

• دالة طباعة حرف واحد putchar() :

ملف التوجيه stdio.

تستخدم لطباعة حرف واحد على الشاشة.

مثال:

```
#include <stdio.h>
#include <conio.h>
void main(){
char ch1,ch2,ch3;
printf("ch1=");
ch1=getchar();
printf("\nch2=");
ch2=getche();
printf("\nch3=");
ch3=getch();
printf("\n");
putchar(ch1);
putchar(ch2);
putchar(ch3);
```

{

فيكون الناتج

ch1= C

ch2= A

=ch3

CA1

بالإضافة إلى الدالة ()putch التابعة لملف التوجيه conio.h وتستخدم أيضاً لطباعة حرف واحد على الشاشة.

- دالة طباعة عبارة حرفية ()puts :

ملف التوجيه stdio.h

تستخدم لطباعة عبارة حرفية string حيث تطبع بدون توصيف شكل المخرجات.

مثال:

```
#include<stdio.h>
void main(){
char name[10]="Ahmed";
puts(name);
puts("Mohamed");
{
```

فيكون الناتج

Ahmed

Mohamed

• دالة إدخال عبارة حرفية ()gets:

ملف التوجيه stdio.h

وتستخدم الدالة ()gets في إدخال عبارة حرفية string.

مثال:

```
#include <stdio.h>
void main(){
char name[10];
printf("\nEnter Your Name:");
gets(name);
puts("welcome");
puts(name);
{
```

فيكون الناتج

Enter Your Name:Samer

welcome

Samer

• دوال تحسين المدخلات و المخرجات:

• دالة مسح الشاشة ()clrscr

ملف التوجيه conio.h

وتستخدم لمسح الشاشة ووضع المؤشر في أول عمود من الصف الأول على الشاشة و تستخدم بالشكل التالي.

•دالة تغيير موضع المؤشر ()gotoxy

ملف التوجيه conio.h

تستخدم لوضع المؤشر في العمود x من الصف Y وتأخذ الصورة التالية:

```
gotoxy(10,30);
```

وتعني انتقال المؤشر إلى العمود رقم ١٠ من الصف ٣٠

•دالة تغير لون الكتابة ()textcolor

ملف التوجيه conio.h

تستخدم لتغير لون الكتابة التي ستطبع بعد الدالة وتأخذ الصيغة:

```
textcolor(color no); or textcolor(color name);
```

حيث يتم تحديد اللون إما برقم اللون أو باسمه ولابد من كتابة اسم اللون بالحروف الكبيرة فقط

والجدول التالي يوضح أكواد الألوان و أسمائها.

اللون	رقم اللون	اسم اللون
أسود	٠	BLACK
أزرق	١	BLUE
أخضر	٢	GREEN

CYAN ٣ سماوي

RED ٤ أحمر

MAGENTA ٥ بنفسجي

BROWN ٦ بني

LIGHTGRAY ٧ رمادي فاتح

DARKGRAY ٨ رمادي غامق

LIGHTBLUE ٩ أزرق فاتح

LIGHTGREEN ١٠ أخضر فاتح

LIGTHCYAN ١١ سماوي فاتح

LIGHTRED ١٢ أحمر فاتح

LIGHTMAGENTA ١٣ بنفسجي فاتح

YELLOW ١٤ أصفر

WHITE ١٥ أبيض

- دالة تغير لون الخلفية textbackground()

ملف التوجيه conio.h

وتستخدم لتغير لون خلفية الكتابة التي ستطبع بعد تحديد لون الخلفية بها وتأخذ الصيغة التالية:

textbackground(color no); or textbackground(color name);

ومعاملات هذه الدالة هي نفس معاملات الدالة السابقة textcolor() مع ملاحظة أن الدالة textbackground() لا تستخدم سوى الألوان من رقم ١ إلى رقم ٧ المذكورين في الجدول السابق.

- **دوال الإدخال و الإخراج التي تستخدم الألوان**

سبق أن أشرنا إلى أن دالة الإخراج printf() ودالة الإدخال scanf() وكذلك باقي الدوال التي تم شرحها قد صممت بحيث تعمل باللون الأبيض على الأسود ولا تتأثر بدوال تغير الألوان فإن هناك مجموعة من الدوال المقابلة للدوال السابقة و التي صممت للتعامل بالألوان المحددة و كلها مسبوقة بالحرف c مثل cgets() cputs() scanf() cprintf() وكلها تابعة لملف التوجيه conio.h

```
#include<stdio.h>
#include<conio.h>
void main(){
textbackground(BLUE);
clrscr();
textcolor(RED);
cprintf("\n This text displayed with Red on Blue
color")
getch();
{
```

فيكون الناتج

This text displayed with Red on Blue color

Loop الدوارات :

for الدوارة

تستخدم لتكرار تنفيذ عملية عدد محدد من المرات و تأخذ الصيغة العامة التالية:

```
For(initial- value; condition; increment)
      statements;
```

حيث:

هي القيمة الابتدائية **initial-value**

هو شرط إنهاء التكرار **condition**

هي قيمة الزيادة الدورية **increment**

طالما أن **(initial-value)**و معناها إبداء العد من القيمة الابتدائية صحيح و مقدار الزيادة كل مرة هو **(condition)**الشرط **increment.**

for تنفيذ أكثر من جملة مع

السابقة كنا نطلب تكرار تنفيذ جملة واحدة عدد محدد **for**في جملة من المرات لكن ما هو العمل إذا كان المطلوب تكرار تنفيذ أكثر من جملة لعدد محدد من المرات. بعبارة أخرى تكرار تنفيذ بلوك كامل داخل البرنامج أكثر من مرة هذا ما سنحاول الإجابة عليه فيما يلي:

لو أردنا تنفيذ أكثر من جملة لعدد محدد من المرات يجب وضع في نهاية } في بداية البلوك المراد تكراره و وضع القوس{القوس البلوك.

مثال

```
#include<stdio.h>
#include<conio.h>
```

```
void main(){
clrscr();
int i;
for(i=0;i<10;i++)
{
printf("\nWelcome ");
printf("with ");
printf("Computer
Science");
}
getch();
}
```

الناتج تكرار العبارة التالية فيكون عشر مرات

Welcome with Computer Science

تغيير معدل الزيادة بالسالب:

في بعض الأحيان نحتاج أن يكون معدل التغيير بالسالب كما في بعض العمليات الحسابية مثلاً ١٠،٩،٨ و هكذا... مثال

```
#include<stdio.h>
#include<conio.h>
void main(){
clrscr();
```

```
int i;
for(i=10;i>0;i--)
{
printf("\n i=%d",i);
}
getch();
}
```

i=10
i=9
.....
i=1

الدوارات المتداخلة باستخدام for :

الدوارات المتداخلة عبارة عن دوارة كبيرة تشتمل بداخلها على دوارة أو أكثر بمعنى أن مجموعة التعليمات الموجودة بالدوارة الداخلية يتكرر تنفيذها طالما لم ينته العداد فإذا انتهى ينتقل التنفيذ إلى الدوارة الخارجية و يتكرر تنفيذ تعليمات الدوارة الخارجية حتى ينتهي العداد المحدد لها. و تشبه فكرة الدورات المتداخلة فكرة عمل عقارب الساعة فتجد عقرب الثواني يدور ٦٠ دورة فيدور عقرب الدقائق بمقدار دقيقة و هكذا.

الدوارة اللانهائية باستخدام for:

ومعناها تكرار تنفيذ الجملة بدون شرط و لا يتوقف التنفيذ حتى يضغط المستخدم Ctrl+c أو Ctrl+break و تأخذ الدوارة صور for. اللانهائية باستخدام for(;;)

- **الدوارة while :**

تستخدم الدوارة while لتكرار تنفيذ جملة أو مجموعة جمل عدد من المرات غير معلوم العدد و إنما يتوقف هذا العدد على شرط موجود بالدوارة و تأخذ الدوارة while الصورة التالية:

```
while(condition)
      statement;
```

و معناها طالما أن الشرط (condition) صحيح نفذ الجملة (statement)

و هي تقوم بتكرار الجملة أو مجموعة الجمل التابعة لها طالما كان شرط التكرار صحيح و عندما يصبح شرط التكرار غير صحيح يتوقف تنفيذ الدوارة.

مثال:

برنامج يقرأ ٥ علامات لطالب ويقوم بحساب المعدل للطالب

```
#include <stdio.h>
#include <conio.h>
void main(){
clrscr();
float avg;
i t sum,x,m;
sum=0;
x=1;
while(x<=5)
{
printf("Enter Mark%d:",x);
scanf("%d",&m);
```

```
sum=sum+m;
x++;
}
avg=sum/5;
printf("Avg=
%.2f",avg);
getch();
}
```

الدوارة for دوارة عددية حيث تعتمد على العداد و ينتهي التكرار فيها بانتهاء عدد مرات التكرار أما الدوارة while فدوارة شرطية أي تعتمد على الشرط الذي يلي الأمر while حيث تتكرر الجمل التي تليها طالما كان الشرط صحيحاً و تنتهي الدوارة بكسر هذا الشرط. و بالتالي الاستخدام الأمثل للدوارة for هو تكرار عملية أكثر من مرة بشرط أن يكون عدد مرات التكرار معلوم و الاستعمال الأمثل للدوارة while هو التكرار بناء على شرط معين.

الدوارة do...while

تستخدم الدوارة do...while لتكرار تنفيذ جملة أو مجموعة جمل أكثر من مرة بناء على شرط معين كما هو الحال مع الدوارة while و لكن الفرق بينهما أن الدوارة while تختبر الشرط أولاً فإذا كان صحيحاً تنفذ الجمل التالية لها و إلاّ فلا، أما الدوارة do ...whileفتنفذ الجمل التالية لها أولاً ثم تختبر الشرط. فإذا كان صحيحاً تعيد التنفيذ و إلاّ توقف التكرار.

و تأخذ الدوارة الصيغة:

```
{
```

```
statement1;
} while (condition);
```

و معناها do أي نفذ الجمل التالية وهي statment1 و ما يليها طالما كان الشرط (condition) صحيحاً.

```
#include <stdio.h>
#include <conio.h>
void main(){
clrscr();
int x=1;
do
{
printf("\n
No=%d\t      uare=%d\tCube=%d",x,x*x,x*x*x);
x++;
}while(x<=9);
getch();
}
```

Branching التفريع :

التفريع يعني تغير مسار البرنامج. و التفريع إما أن يكون مشروط كجملة if أو غير مشروط كجملة goto.

- التفريع المشروط:

جملة الشرط if :

تستخدم كلمة لتنفيذ جملة أو أكثر حسب شرط معين و أبسط صورة if هي: لجملة

```
if (condition)
    statment;
```

نفذ الجملة التالية أما إذا لم (condition)و معناها إذا تحقق الشرط يتحقق الشرط فلا تنفذ هذه الجملة و انتقل إلى التي تليها.
ملاحظة: إذا كان هناك أكثر من جملة نريد تنفيذها مع لا بد من فتح قوس{ قبل مجموعة الجمل و القوس} في آخر الجمل كما يلي:

```
if (condition)
  {
statement1;
statement2;
  }
```


جمل الشرطية المتداخلة :if

فتأخذ الشكل التالي:ifيمكن أن تتداخل جمل

```
if
(condition)
                    if
(condition)
    if (condition)
```

و هذا معناه إذا تحقق الشرط الأول انظر إلى الشرط الثاني.. و هكذا.

if...else الجملة الشرطية

تستخدم لتنفيذ أحد اختياريين و تأخذ الصورة التالية:

```
if
(condition)
    {
      st      ement1
    }
else
    {
        statement2
   }
```

صحيح نفذ الجملة الأولى (condition)و معناها إذا كان الشرط
statment1 و إلاّ نفذ الجملة الثانية statment2
تستخدم لتحديد اختيار واحد من if...elseو هذا يعني أن تركيب
معاً كما يحدث مع جملةif اختيارين و لا يمكن تنفيذ الاختيارين

if...else if الجملة الشرطية

لتنفيذ خيار من مجموعة خيارات كمقارنة رقمين مثلاً حيث يكون الرقم الأول أكبر من أو يساوي أو أقل من الرقم الثاني. يوجد طريقتين،
و في كل جملة نضع أحد الشروط ifالطريقة الأولى استخدام ثلاث جمل الثلاثة كالآتي:

```
i=5;
if(i<5)
      printf("i     less
than                  5");
if(i=5)
   printf("i  equal  to
```

```
5");
if(i>5)
    printf("i  greater
than 5");
```

الثلاثة حتى و ifو يعيب تلك الطريقة أن البرنامج سيقوم باختبار شروط الثانية فهو لا بد أن يختبر جملة ifإن كان الشرط قد تحقق في جملة الثالثة لأن كل جملة من جمل الشرط مستقلة بنفسها و يجري تنفيذها if على حدة مما يستهلك وقتاً في اختبار جمل شرطية لا داعي لاختبارها حيث نفذت إحداها بالفعل.

و الطريقة الثانية تستخدم لتلافي ذلك العيب و فيها نستخدم الجملة الشرطية if...else if: و صيغتها كالآتي:

```
if(condition)
          statement1;
else    if(condition)
          statement2;
else    if(condition)
    statement3;
```

و هذا معناه تحديد اختيار من عدة اختيارات.

```
i=5;
if(i<5)
printf("i less than
5");
else if(i=5)
printf("i equal to
5");
```

```
else if(i>5)
printf("i greater
than 5");
```

مثال:

برنامج آلة حاسبة بسيطة

```
#include<stdio.h>
#include <conio.h>
void main(){
clrscr();
float num1,num2;
char op,ch;
do{
printf("\n Type
num1,op,num2\n");
scanf("%f %c
%f",&num1,&op,&num2);
if(op=='+')

printf("sum=%.2f",num1+num2);
else if(op=='-')
printf("sub=%.2f",num1-
num2);
else if(op=='*')

printf("Mult=%.2f",num1*num2);
```

```
else if(op=='/')

printf("div=%.2f",num1/num2);
printf("\n again(y) or
press any key ");
}while((ch=getch())=='y');
}
```

إن استخدام جملة اف في حالة تعدد الاختيارات لأكثر من اختيارين يمثل عبئاً على المبرمج في تتبع خطوات البرنامج و يسبب بطئاً نسبياً في تنفيذ البرنامج لذا استخدمنا الجملة الشرطية if..else if و يمكن استعمال switch..case كبديل لجملة if..else if و هي طريقة أسهل للتفريع كما سنرى و تستخدم بالصيغة التالية:

```
ch=getch();
switch(ch)
{
case'1':
staement1;
statement2:
...
break;
case'2':
statement1;
statement2;
...
break;
default:
```

```
statement1;
statement2;
…
}
```

مثال:

```
#include<stdio.h>
#include <conio.h>
void main(){
 float num1,num2;
 char op,ch;
 do{
  clrscr();
  printf("\n Type
num1,op,num2\n");
  scanf("%f %c
%f",&num1,&op,&num2);
  switch(op)
  {
   case'+':

printf("sum=%.2f",num1+num2);
   break;
   case'-':

printf("sub=%.2f",num1-num2);
   break;
```

```
case'*':

printf("Mult=%.2f",num1*num2);
break;
case'/':

printf("div=%.2f",num1/num2);
break;
default:
printf("\n unknown
operator..");
}
printf("\n again(y) or
press any key ");
}while((ch=getch())=='y');
}
```

المؤثر الشرطي Conditional operator

هذا المؤثر يقوم مقام جملة ، و يأخذ الشكل التالي:

```
var=(condition)?exp1:exp2;
```

و معناه

```
if condition is true then
var=exp1
if condition is false then
var=exp2
```

مثال

```
Pay=(age>18)? 3.5:2.5
```

الشرط التالي هذا الشرط يساوي

```
if (age>18)
  Pay=3.5;
else
  Pay=2.5
```

- التفريع غير المشروط:

التفريع غير المشروط معناه الانتقال إلى مكان محدد داخل البرنامج بدون شرط و تقوم جملة goto بهذا الغرض و تأخذ الشكل العام التالي:

goto Label

حيث Label يشير إلى المكان المطلوب الانتقال إليه و لا ننصح باستخدام هذه الجملة لأنها تستخدم بكثرة مع اللغات الغير تركيبية مثل لغة بيسك أما في حالة اللغة التركيبية لغة C فيفضّل استخدام الدوال لتغير مسار تنفيذ البرنامج.

المصفوفات ARRAYS

معنى المصفوفات

تنقسم البيانات إلى بيانات حرفية(char) وبيانات رقمية(int) وبيانات حقيقية (float)وتسمى هذه الأنواع (int, float, char) بالأنواع الرئيسية للبيانات، حيث لا يمكن تجزئتها أقل من ذلك.

ولكن هناك أنواع أخرى من البيانات تسمى بالأنواع المشتقة (types

Drived data) من هذه الأنواع المصفوفات Arrays. تعرف المصفوفة بأنها مجموعة من العناصر تنتمي إلى نوع واحد. ويخصص لها اسم واحد وتنقسم المصفوفات إلى مصفوفات ذات بعد واحد ومصفوفات ذات بعدين.

المصفوفة ذات البعد الواحد مثل:

A= [3 4 5 7 9]

وتسمى مصفوفة ذات بعد واحد لأنها تتكون من صف واحد أو عمود واحد، وفيها حرف A هو اسم المصفوفة، والأرقام هي عناصر المصفوفة ويتم الإشارة إلى كل عنصر برقم العنصر أي بترتيبه داخل المصفوفة على أن يبدأ العد بالرقم صفر كما يلي:

العنصر A[0] يساوي ٣ والعنصر A[1] يساوي ٤ و العنصر A [2] يساوي ٥.

المصفوفة ذات البعدين تأخذ الشكل التالي:

وتسمى المصفوفة 3x3 أي ٣ صفوف و٣ أعمدة ويتم الإشارة إلى عناصر المصفوفة برقم الصف ورقم العمود الذي يقع عندهما العنصر كما يلي:

العنصر C[0][0] يساوي ٥

العنصر C[0][1] يساوي ٤

العنصر C[0][2] يساوي ٢

العنصر C[2][1] يساوي ٩

والخلاصة أن المصفوفة هي مجموعة من العناصر سواء ذات بعد واحد

أو بعدين بشرط أن تكون جميع العناصر من نوع واحد وفيما يلي سنوضح كيفية الإعلان عن المصفوفة وكيفية التعامل مع عناصرها.

•المصفوفة ذات البعد الواحد:

البرنامج التالي يوضح التعامل مع المصفوفة ذات البعد الواحد و فيه يتم الإعلان عن المصفوفة و استقبال عناصر المصفوفة من المستخدم و إضافة قيمة صحيحة إلى كل عنصر من عناصر المصفوفة ثم طباعة عناصر المصفوفة كما يتضح ذلك من نتيجة التنفيذ.

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
int A[10];
int i;
for (i=0;i<10;i++)
{
printf ("\n
A[%d]=",i);
scanf("%d",& A
[i]);
A[i]=A[i]+5;
}
for (i=0;i<10;i++)
printf ("\n
A[%d]=%d", i, A [i]);
```

```
getch();
    }
```

ملاحظة:

١. لا بد من استعمال الدوارة for مع المصفوفات.
٢. يشار لأول عنصر في المصفوفة بالرقم صفر هكذا A[٠].

إعطاء قيم ابتدائية لعناصر المصفوفة:

من الممكن الإعلان عن المتغير وإعطائه قيمة ابتدائية بالشكل التالي:int A=5 و هذا الإعلان عن متغير صحيح و في نفس الوقت إعطاءه قيمة ابتدائية.
و بنفس الأسلوب يمكن الإعلان عن المصفوفة وإعطائها قيم ابتدائية كما يلي:

```
int   A   [3]   =   {5,7,9};
char   name   [10]   =
{'c','b','t','r',----};
```

و هذا معناه إعطاء قيم ابتدائية لعناصر المصفوفة و هو الأفضل كلما استطعت ذلك حتى لا يقوم البرنامج بتخزين قيم عشوائية من الذاكرة في عناصر المصفوفة و حتى لا تطبع قيم ليس لها معنى.
المصفوفة غير محددة العدد:المقصود بها هو عدم تحديد عدد العناصر في حالة الإعلان و تأخذ الصورة التالية:

```
int   A   []=   {3,4,5};
Char name [] ="abdef";
```

و تحديد عدد عناصر هذه المصفوفة في هذه الحالة يتم من خلال المترجم عن طريق عد العناصر في الطرف الأيمن و حجز مصفوفة بهذا العدد.

لا يصلح إلاّ إذا كنت ستعطي عناصر المصفوفة قيم ابتدائية و لكن لا يصح أن تعلن عن مصفوفة غير محددة العدد ثم تستعملها في استقبال قيم من المستخدم فمثلاً لا يصح أن تقول int a[]; ثم تستقبل عناصر المصفوفة من المستخدم.

• **المصفوفة ذات البعدين:**

هي المصفوفة التي ترتب عناصرها في شكل صفوف و أعمدة و يتم الإعلان عنها بالشكل التالي:

```
int A [5] [10];
```

و هذا معناه أن المصفوفة A مصفوفة ذات بعدين، ٥ صفوف و ١٠ أعمدة و يتم الإشارة إلى العنصر برقم الصف و رقم العمود.

و يجب الانتباه إلى أنه عندما تستخدم مصفوفة لا بد من استعمال الدوارة for و يتضح ذلك من المثال الذي ذكرناه في المصفوفة ذات البعد الواحد و أما في حالة المصفوفة ذات البعدين فلا بد من استعمال ما يسمى بالدوارات المتداخلة **Nested loops**.

و هذا ما نراه من خلال البرنامج التالي حيث يقوم باستقبال مجموعة قيم و يخزنها في مصفوفة ذات بعدين ثم يقوم بطباعة هذه القيم في شكل مصفوفة ذات بعدين.

```
#include<stdio.h>
#include<conio.h>
void main ()
{
clrscr();
int x, y;
```

```
int A[3][4];
for(x=0; x<3; x++)
{
printf ("\n");
for(y=0; y<4; y++)
{
printf ("\t A[%d]
[%d]=", x, y);
scanf ("%d", &
A[x] [y]);
}
}
clrscr();
for(x=0; x<3; x++)
{
printf ("\n");
for(y=0; y<4; y++)
printf ("\t %d", A
[x] [y]);
}
getch();
}
```


إعطاء قيم ابتدائية للمصفوفة ذات البعدين:

كما يمكن إعطاء قيم ابتدائية للمصفوفة ذات البعد الواحد يمكن كذلك إعطاء قيم ابتدائية للمصفوفة ذات البعدين و يكون بالشكل التالي:

```
int
```

A[3][4]={{4,5,7,8},{3,2,1,0},{12,10,7,88}};

إنشاء الدوال و الماكرو Functions and macros

• **المقصود بالدالة:**

الدوال التي استخدمناها في الفصول السابقة مثل (scanf or print) دوال مبيتة في لغة C وهي دوال عامة يستطيع أي مبرمج استخدامها. من مزايا لغة C المرونة في الاستخدام ولذلك يمكن إنشاء دوال مثل الدوال القياسية الموجودة في صلب اللغة لتؤدي وظائف مختلفة أو متشابهة والدالة عبارة عن برنامج صغير (أو مجموعة تعليمات تؤدي غرض معين) يخصص لهذا البرنامج اسم ويتم استدعائه داخل الدالة الرئيسية(main) بهذا الاسم.

ويحقق استخدام الدوال مزايا عديدة منها:

- **عدم تكرار التعليمات داخل البرنامج حيث يتم إنشاء الدالة مرة واحدة ثم يتم استدعائها أكثر من مرة عند الحاجة إليها.**
- **باستخدام الدوال يصبح البرنامج أكثر وضوحًا حيث يأخذ البرنامج الشكل التركيبي فيصبح بالشكل الآتي:**

```
#include<filename.h>
        Functions
     declaratiobs,
     void main ()
```

```
{
Function1-calling ();
Function2-calling ();
.......
.......
}
Function1-
defination()
{
.......
}
Function2-defination
()
{
.......
}
```


وبهذا يصبح البرنامج كما ترى أسهل للفهم حيث يتكون من الدالة الرئيسية ومن داخلها يتم استدعاء مجموعة من الدوال وبالتالي يكفي أن تفهم عمل كل دالة لفهم البرنامج كله.

- يمكن للمبرمج المتمرس إنشاء مكتبة دوال خاصة توفر عليه إعادة كتابة البرامج في كل مرة يحتاج إليها.

•أنواع الدوال Functions types:

وعرفنا أن كلمة **void** هي أحد أنواع الدوال وهناك أنواع أخرى من الدوال ونوضحها فيما يلي:

دوال تعيد قيمة صحيحة **int function**.

دوال تعيد قيمة حقيقية **float function**.

دوال تعيد عبارة حرفية**string function**.

دوال تعيد حرف واحد **char function**.

دوال لا تعيد أي قيمة **void function**.
دوال تعيد قيمة من نوع **structure** وتسمى **struct function**
مثال

```
#include <stdio.h>
#include <conio.h>
int sum(int a,int b);
int sub(int a,int b);
void main()
{
clrscr();
int x,y;
printf("Enter num1:");
scanf("%d",&x);
printf("Enter num2:");
scanf("%d",&y);
printf("\nThe sum & sub for num1,num2:");
```

```
printf("\n\t\tsum=%d\tsub=%d",sum(x,y),sub(x,y));
getch();
}
int sum(int a,int b)
{
return (a+b);
}
int sub(int a,int b)
{
return (a-b);
}
```


•الماكرو(MACROS)

ما المقصود بالماكرو؟

هو مجموعة تعليمات تؤدي غرض معين ويشبه إلى حد كبير الدالة، ويتم إنشائه مرة واحدة وبعد ذلك يمكن استدعائه كلما احتجت إليه.

وقبل أن نسأل ما الفرق بينه وبين الدالة دعنا نرى كيفية إنشائه واستعماله ثم نناقش بعد ذلك الفرق ثم نوضح متى نستخدم الماكرو ومتى نستخدم الدالة.

يتم ذلك باستعمال الكلمة #define وهذه الكلمة تسمى directive أو preprocessor ومعناها توجيه.

ولإنشاء الماكرو تستخدم الصورة التالية:

#define macro line

و هي عبارة عن تعريف طرف بطرف مثل #define A 5 ومعناها عرف المتغير A بالقيمة ٥.

مثال:يوضح البرنامج كيفية الإعلان عن الماكرو و كيفية استعماله

```
#define sum(a,b) a+b
#define mul(x,y) x*y
#include <stdio.h>
#include <conio.h>
void main()
{
clrscr();
int v1=5,v2=10;

printf("\n\tsum(v1,v2)=%d",sum(v1,v2));

printf("\n\tmul(v1,v2)=%d",mul(v1,v2));
getch();
}
```

فيكون الناتج

sum(v1,v2)=15

mul(v1,v2)=50

السجلات Structures

و الحاجة إلى استعماله:structure معنى السجل

من أهم التطبيقات في عالم البرامج تطبيقات قواعد البيانات فمثلاً قاعدة بيانات موظفين تمثل بيانات الموظفين في شكل سجلات كل سجل يتكون من مجموعة من حقول و لو أن لك خبرة بأحد برامج قواعد البيانات مثل dbase فستعرف أن الملف ينقسم إلى سجلات records و السجل ينقسم إلى حقول fields و كذلك و دائماً نحتاج للتعامل مع السجل كوحدة مع الحقول كوحدة . و تستخدم لغة C كلمة struct بنفس المفهوم الذي تستخدمه لغات البرمجة الأخرى لكلمة Record

استعمال السجل:

هناك خطوات تتبع للتعامل مع السجل و هي إنشاء السجل (تركيب السجل) و تحديد الحقول المطلوبة ثم الإعلان عن متغير من نوع هذا السجل ثم التعامل مع حقول هذا السجل. و البرنامج التالي يشتمل على هذه الخطوات:

```
#include <stdio.h>
#include <conio.h>
void main ()
{
clrscr();
struct data
{
int num;
char stat;
};
```

```
struct data stud;
stud.num = 5;
stud. stat='t';
printf ("\n stud.num =%d,
stud.stat=%c", stud.num,
stud. stat);
getch();
}
```

فيكون الناتج

```
strud.num =5, stud.stat=t
```

وضع محتويات سجل في آخر:

رأينا من قبل إمكانية مساواة متغيرين من نوع واحد و ذلك لوضع قيمة المتغير الأول في المتغير الثاني و يمكن تحقيق ذلك مع السجلات بحيث يمكن مساواة متغير من نوع سجل مع آخر و بالتالي يتم مساواة قيم جميع العناصر بين السجلين بشرط أن يكون السجلين من نفس النوع، و البرنامج التالي يوضح ذلك:

```
#include <stdio.h>
#include <conio.h>
void main ()
{
struct data
{
int no;
char name[10];
};
```

```
struct data stud1, stud2;
clrscr();
printf("stud1.no = ");
scanf("%d",&stud1.no);
printf("stud1.name = ");
scanf("%s",&stud1.name);
stud2 = stud1;
printf ("\n stud1.no =%d\t
stud1.name =%s",stud1. no,
stud1.name);
printf ("\n stud2.no =%d\t
stud2. name =%s",stud2.no,
stud2.name);
getch();
}
```

فيكون الناتج

```
stud1.no= 5        stud1.name=
tamer

stud2.no= 5        stud2.name=
tamer
```


السجلات المتداخلةNested Structures

شرحنا أن السجل هو مجموعة من العناصر أياً كان نوع هذه العناصر و بالتالي يمكن أن تكون العناصر أو بعضها سجلات و هذا ما يسمى بالسجلات المتداخلة و البرنامج التالي يوضح كيف يكون السجل عنصر

في سجل آخر و كيفية التعامل مع عناصر السجلات في هذه الحالة.

```
#include <stdio.h>
#include <conio.h>
void main()
{
struct person{
int no;
char name[10];
};
struct group{
struct person P1;
struct person P2;
int code;
};
struct group G1;
clrscr();
printf("\n\n G1.P1.no= ");
scanf("%d",&G1.P1.no);
printf("\n G1.P1.name= ");
scanf("%s",G1.P1.name);
printf("\n G1.code = ");
scanf("%d",&G1.code);
G1.P2=G1.P1;
clrscr();
printf("\n\n The data of
Groups :\n\t");
```

```
printf("\n\t G1.P1.no=%d",G1.P1.no);
printf("\n\t G1.P1.name=%s",G1.P1.name);
printf("\n\t G1.code=%d",G1.code);
printf("\n\t G1.P2.no=%d",G1.P2.no);
printf("\n\t G1.P2.name=%s",G1.P2.name);
getch();
}
```

فيكون الناتج

```
G1.P1.no=          10
G1.P1.name=        Samer
G1.code =199

The data of Groups :
G1.P1.no=10
G1.P1.name= Samer
G1.code=199
G1.P2.no=10
G1.P2.name= Samer
```

المؤشر pointer

المؤشر هو نوع من أنواع البيانات ويعرف بأنه متغير يحتفظ (يخزن به)

بعنوان مكان في الذاكرة.

من المعلوم أن كل مكان في الذاكرة له عنوان والجهاز يتعامل مع هذا المكان بالعنوان المحدد له ونحن بطريقة غير مباشرة نتعامل مع هذا العنوان، فمثلاً هذا الإعلان int a=5; معناه احجز مكان في الذاكرة (RAM) حجمه ٢ بايت(حجم int) واجعل اسمه a وضع فيه القيمة ٥. وبالتالي كلما تعاملنا مع المتغير a فنحن نتعامل مع القيمة المخزنة فيه وليس العنوان المخصص لهذه القيمة. هذا عن الإعلان العادي، فماذا عن الإعلان المؤشر (pointer).

- الإعلان عن المؤشر pointer:

يتم الإعلان عن المؤشر إلى أي متغير من أنواع البيانات بنفس الطريقة التي نعلن بها عن البيانات العادية وهي تحديد نوع البيانات ثم اسم المتغير ولكن الفرق بين الإعلان عن المتغير والإعلان عن المؤشر أن اسم المتغير يجب أن يسبق بالعلامة* ليدل على أنه مؤشر، أي أن العلامة* تجعل المتغير مؤشر. فمثلاً للإعلان عن مؤشر من نوع صحيح نكتب الصورة التالية: int*p; وكما ترى

ليس هناك جديد غير اسم المتغير سبق بالعلامة*

وماذا يعنى هذا الإعلان؟

يعنى أن المتغير p أصبح مؤشر إلى مساحة في الذاكرة مقدارها٢ بايت مع الاحتفاظ بعنوان هذا المكان في المتغير P.

هل لاحظت كلمة عنوان هذا ما يهمنا، وكلما أردنا أن نتعامل مع هذه القيمة تعاملنا عن طريق العنوان أي بدلاً من أن نتعامل نحن مع القيمة ونترك الجهاز يتعامل مع العنوان بهذا الأسلوب نستطيع أن نتعامل مباشرة مع عنوان المكان مما يعطينا القدرة على عمليات كثيرة منها التعامل مع مخارج الجهاز مثل مخرج آلة الطباعة حيث أن لمخرج الطابعة عنوان فنستطيع أخذ هذا العنوان وتخزينه في متغير ثم التعامل مع هذا المتغير كما نشاء وكذلك الكتابة في ذاكرة العرض مباشرة وهكذا.

يمكن للمؤشر أن يشير إلى أي نوع من أنواع البيانات حسب الإعلان.

شرحنا كيف يتم الإعلان عن مؤشر يشير إلى قيمة صحيحة فكيف يكون الإعلان عن مؤشر يشير إلى قيمة حقيقية (pointer to float).
يكون ذلك بالصورة التالية: float *k;
ومعناه أن احجز مكان في الذاكرة مقدارها ٤ بايت وخزن عنوان هذا المكان في المتغير k الذي يحتفظ بهذا العنوان.
إذن طريقة الإعلان عن مؤشر إلى أي نوع من أنواع البيانات هي نفس الطريقة المستخدمة للإعلان عن المتغيرات غير أننا نسبق المتغير بالعلامة* وهذا يعني أنه مؤشر إلى هذا النوع.

• مزايا استخدام المؤشرات pointer:
يحقق استعمال المؤشرات فوائد كثيرة منها:
• إعادة أكثر من قيمة من الدوال.
• التعامل مع المصفوفات و الحرفية وتمريرها إلى الدوال بشكل أفضل.
• إنشاء أنواع أكثر قوة من البيانات.
• التعامل مع الجهاز ومكوناته وعناوين مداخل ومخارج الجهاز.

• إعادة أكثر من قيمة من الدوال:
من الفوائد المشهورة للمؤشرات استخدامها في إعادة أكثر من قيمة من الدالة.
فما معنى هذا؟
شرحنا الدوال والتعامل معها وكيفية إعادة قيمة من الدالة لاحظنا في الأمثلة التي استخدمناها أننا استخدمنا كلمةreturn مرة واحدة مع كل دالة وهذا معناه عدم إمكانية إعادة أكثر من قيمة من الدالة.
فلو فرضنا أن لدينا مجموعة عمليات وأردنا إنشاء دالة لهذه العمليات وأنشأنا الدالة وتم حساب نتائج العمليات ووضعت هذه النتائج في متغيرات وأردنا إعادة هذه القيم إلى الدالة الرئيسية ، هنا تظهر المشكلة. وهي أننا

لا نستطيع استعمال أكثر من كلمة return وكلمة return لا تعيد إلا قيمة واحدة أما في حالة استخدام المؤشرات فيمكننا إعادة أكثر من قيمة.

عمل المؤشرات و يبين ميزة

مثال: برنامج يوضح عمل المؤشر:

```
#include<stdio.h>
#include<conio.h>
void get2(int *xx, int *yy);
void main(){
int x=5,y=10,*p1,*p2;
p1=&x,p2=&y
get2(p1,p2);
clrscr();
printf("\nfirst no:is
%d\tsecond no:is %d",x,y);
getch();
}
void get2(int *xx,int *yy)
{
*xx+=5;
*yy+=10;
```

}

الناتج فيكون

first no:is 10 second no:is 20

....

وسلام عليكم ورحمة الله و بركاته...............

مصدر معلومات الكتاب:

١-من خلال دراستي الجامعية.

٢-الانترنت.

تم بحمد الله الانتهاء من كتابة هذه النوتة وإن شاء الله هي خالية من الأخطاء وجل من لا يسهى

ونرجو منكم الدعاء لنا في ظهر الغيب ولله المبتغى من وراء القصد.



أخوكم في الله

بشير عبده فارع محمد

basheer2010.55@gmail.com